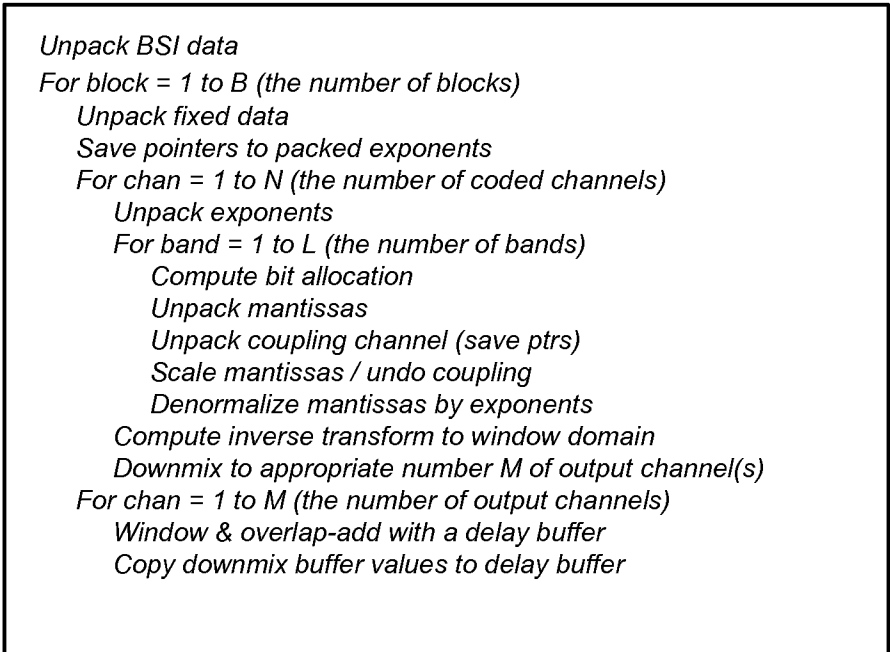


1/13



```
graph TD; A[Unpack BSI data] --> B[For block = 1 to B (the number of blocks)]; B --> C[Unpack fixed data]; C --> D[Save pointers to packed exponents]; D --> E[For chan = 1 to N (the number of coded channels)]; E --> F[Unpack exponents]; F --> G[For band = 1 to L (the number of bands)]; G --> H[Compute bit allocation]; H --> I[Unpack mantissas]; I --> J[Unpack coupling channel (save ptrs)]; J --> K[Scale mantissas / undo coupling]; K --> L[Denormalize mantissas by exponents]; L --> M[Compute inverse transform to window domain]; M --> N[Downmix to appropriate number M of output channel(s)]; N --> O[For chan = 1 to M (the number of output channels)]; O --> P[Window & overlap-add with a delay buffer]; P --> Q[Copy downmix buffer values to delay buffer];
```

Unpack BSI data
For block = 1 to B (the number of blocks)
Unpack fixed data
Save pointers to packed exponents
For chan = 1 to N (the number of coded channels)
Unpack exponents
For band = 1 to L (the number of bands)
Compute bit allocation
Unpack mantissas
Unpack coupling channel (save ptrs)
Scale mantissas / undo coupling
Denormalize mantissas by exponents
Compute inverse transform to window domain
Downmix to appropriate number M of output channel(s)
For chan = 1 to M (the number of output channels)
Window & overlap-add with a delay buffer
Copy downmix buffer values to delay buffer

100

FIG. 1
(Prior Art)

2/13

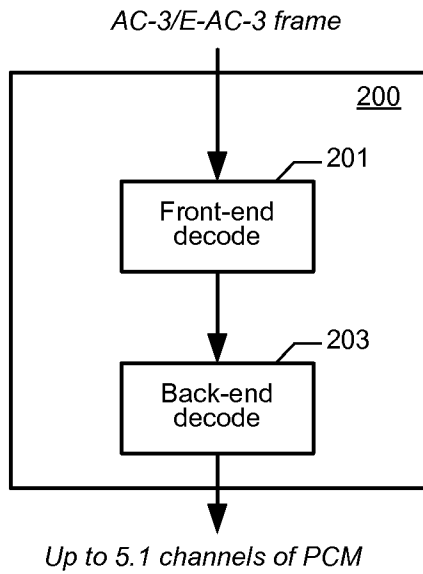


FIG. 2A

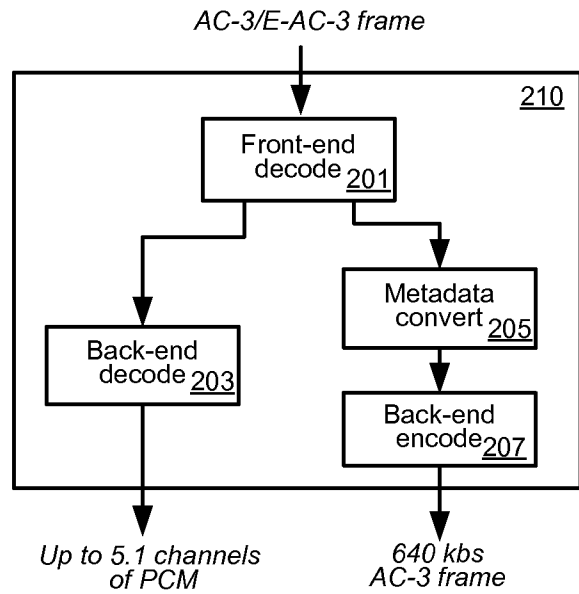


FIG. 2B

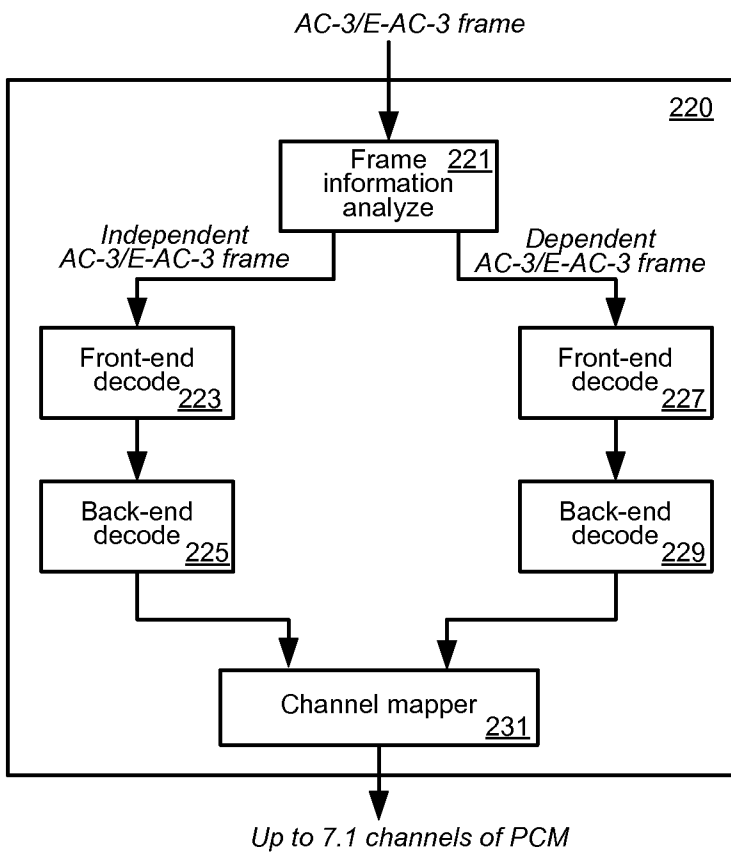


FIG. 2C

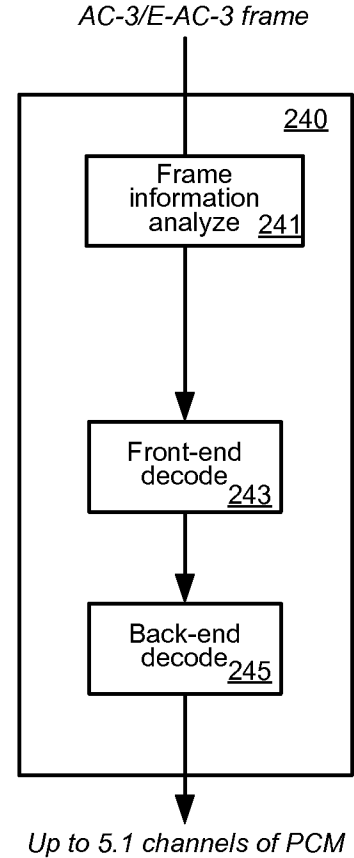


FIG. 2D

3/13

Front-end decode

300

```

/* First pass front-end decode */
For block = 0 to B-1 (B=number of blocks)
{
    Unpack fixed data
    For chan = 0 to N-1 (N=number of coded channels)
    {
        Save bitstream pointer to packed exponents
        Unpack exponents
        Save bitstream pointer to packed mantissas
        Compute bit allocation
        Skip Mantissas based on bit allocation
    }
}

/* Second pass front-end decode */
For channel = 0 to N-1 (N=number of coded channels)
{
    For block = 0 to B-1 (B=number of blocks)
    {
        /* unpack */
        Load saved bitstream pointer to packed exponents
        Unpack exponents
        Compute bit allocation
        Load saved bitstream pointer to packed mantissas
        Unpack mantissas
        /* decode */
        Perform standard/enhanced (amplitude-only) decoupling
        Generate spectral extension band
    }
    Transfer exponent and mantissa data from internal to external memory
}

```

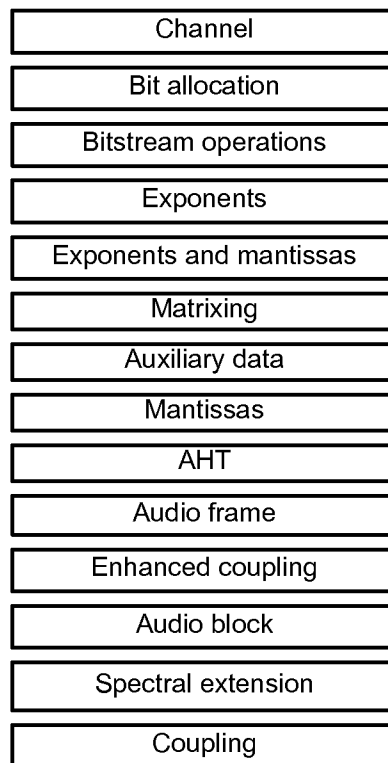


FIG. 3

4/13

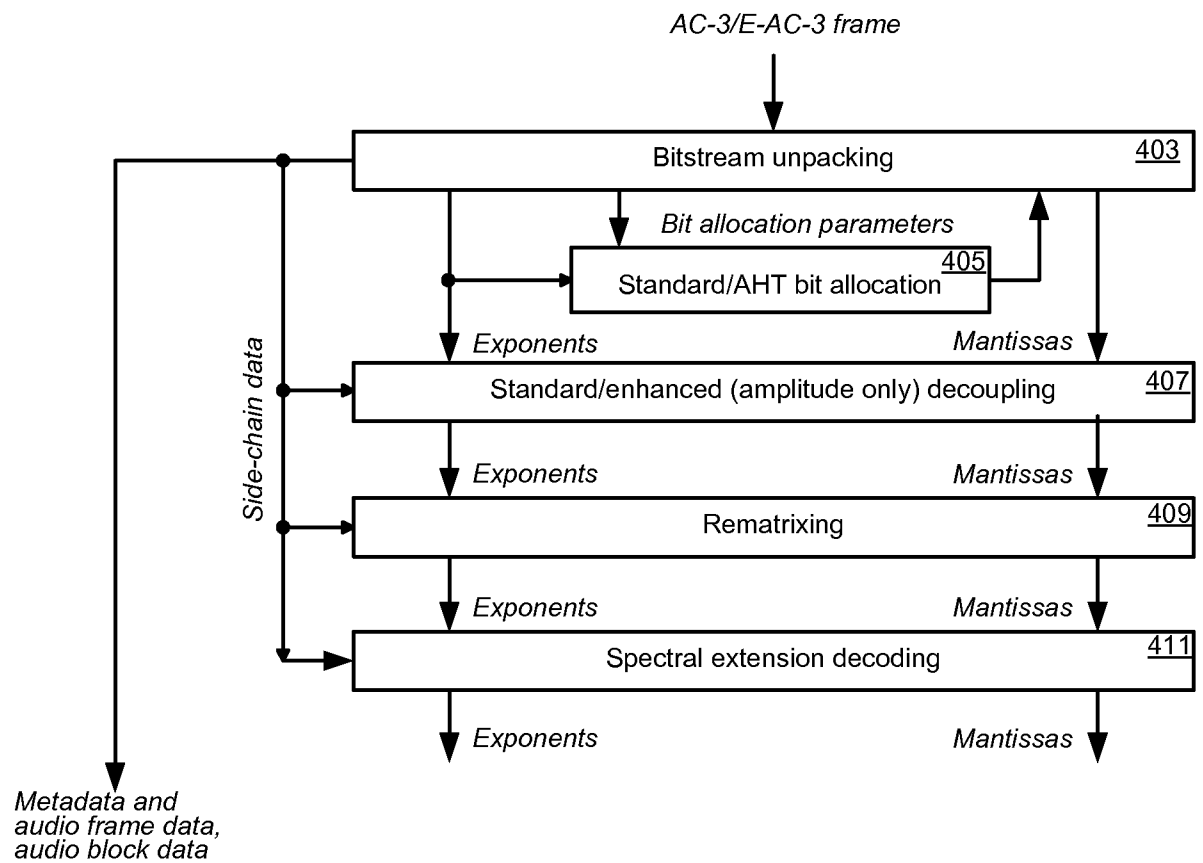


FIG. 4

5/13

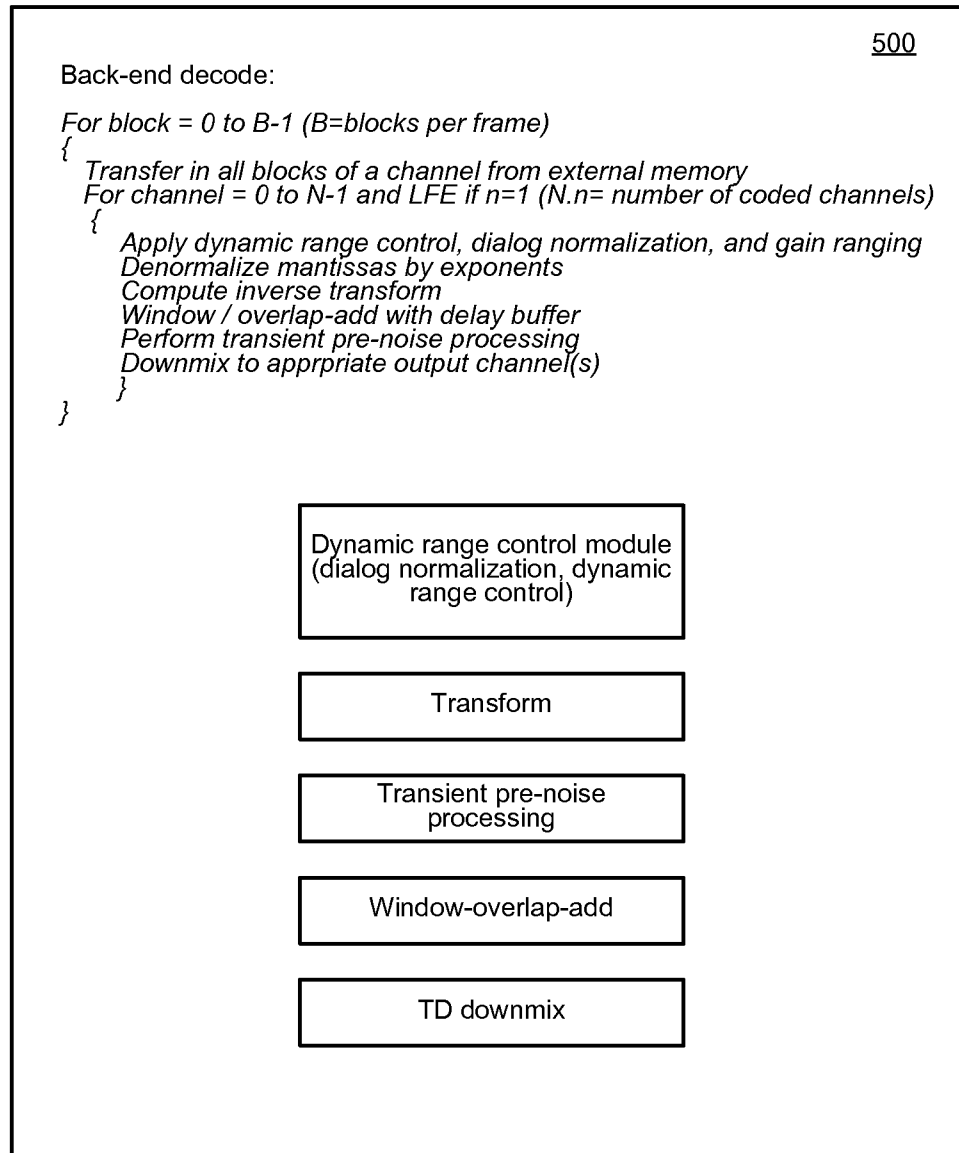


FIG. 5A

6/13

520

```

% Back-end decode:
For block = 0 to B-1 (B=blocks per frame)
{
  Transfer in all blocks of a channel from external memory
  Ascertain if FD downmixing or TD downmixing
  If FD downmixing:
  {
    For channel = 0 to N-1 and LFE if n=1
    {
      Apply dynamic range control, dialog normalization, but disable gain ranging
      Denormalize mantissas by exponents
      FD downmix
      Process differently transition block following TD downmixed block
      Deal with any fading out channels
    }
  }
  Else % TD downmixing
  {
    For channel = 0 to N-1 and LFE if n=1 (BUT only for channels in downmix)
    {
      Process differently transition block following FD downmixed block
      Apply dynamic range control, dialog normalization, but disable gain ranging
      Denormalize mantissas by exponents
    }
  }
  For channel = 0 to N-1 and LFE if n=1 (BUT only for channels in downmix; N=M if FD
  downmix)
  {
    Compute inverse transform
    Window / overlap-add with delay buffer
    If TD downmix
    {
      Perform transient pre-noise processing
      Downmix to appropriate output channel(s)
    }
  }
}

```

Dynamic range control module
(dialog normalization, dynamic
range control)

Downmix method selection
module

FD downmix transition logic
module

FD Downmix
(includes TD downmix
transition logic module)

Transform

Transient pre-noise
processing

Window-overlap-add

TD downmix

FIG. 5B

7/13

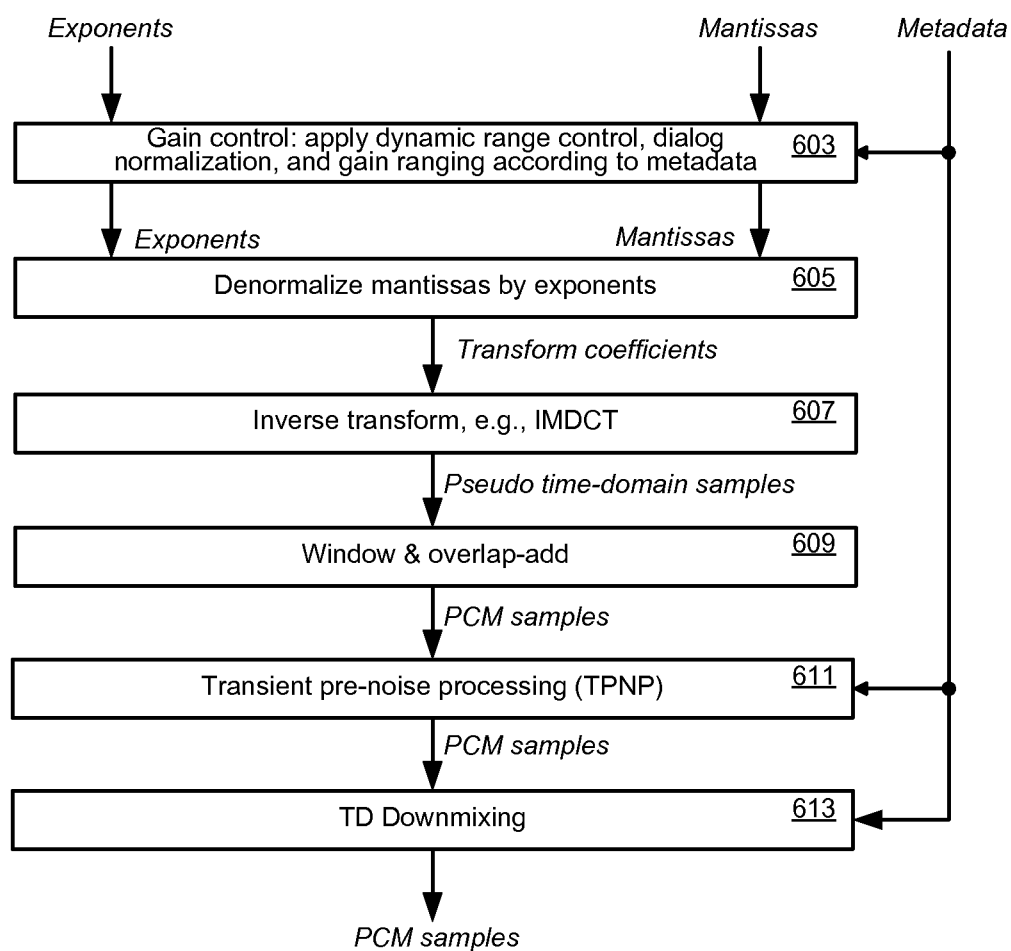


FIG. 6

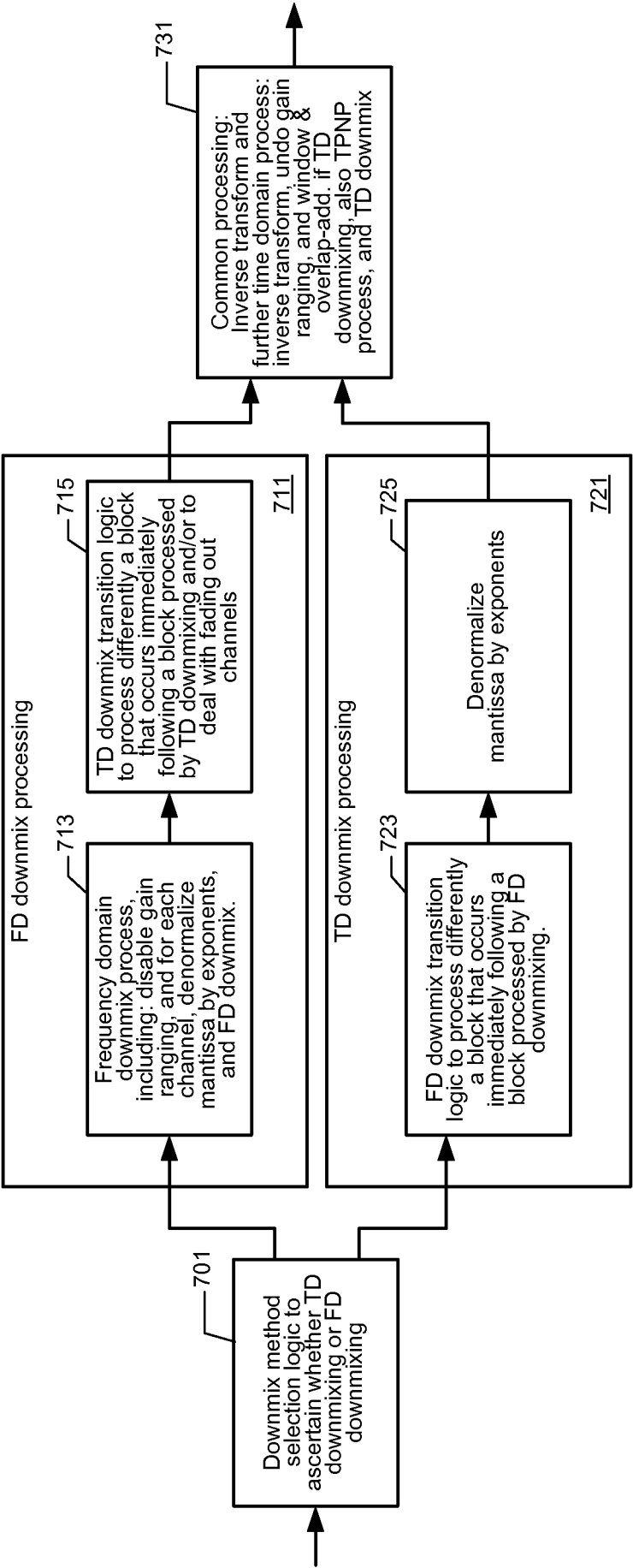


FIG. 7

9/13

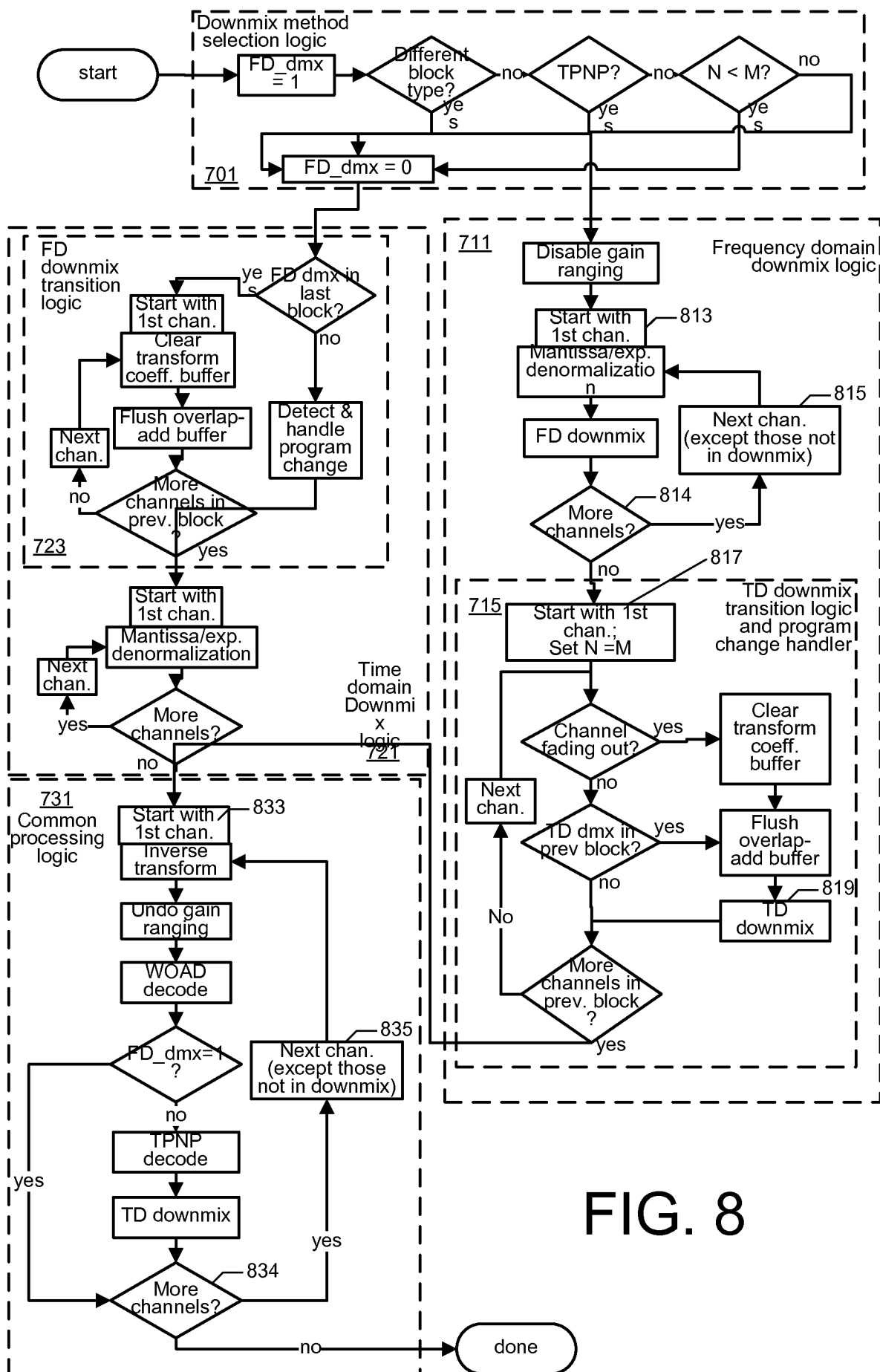
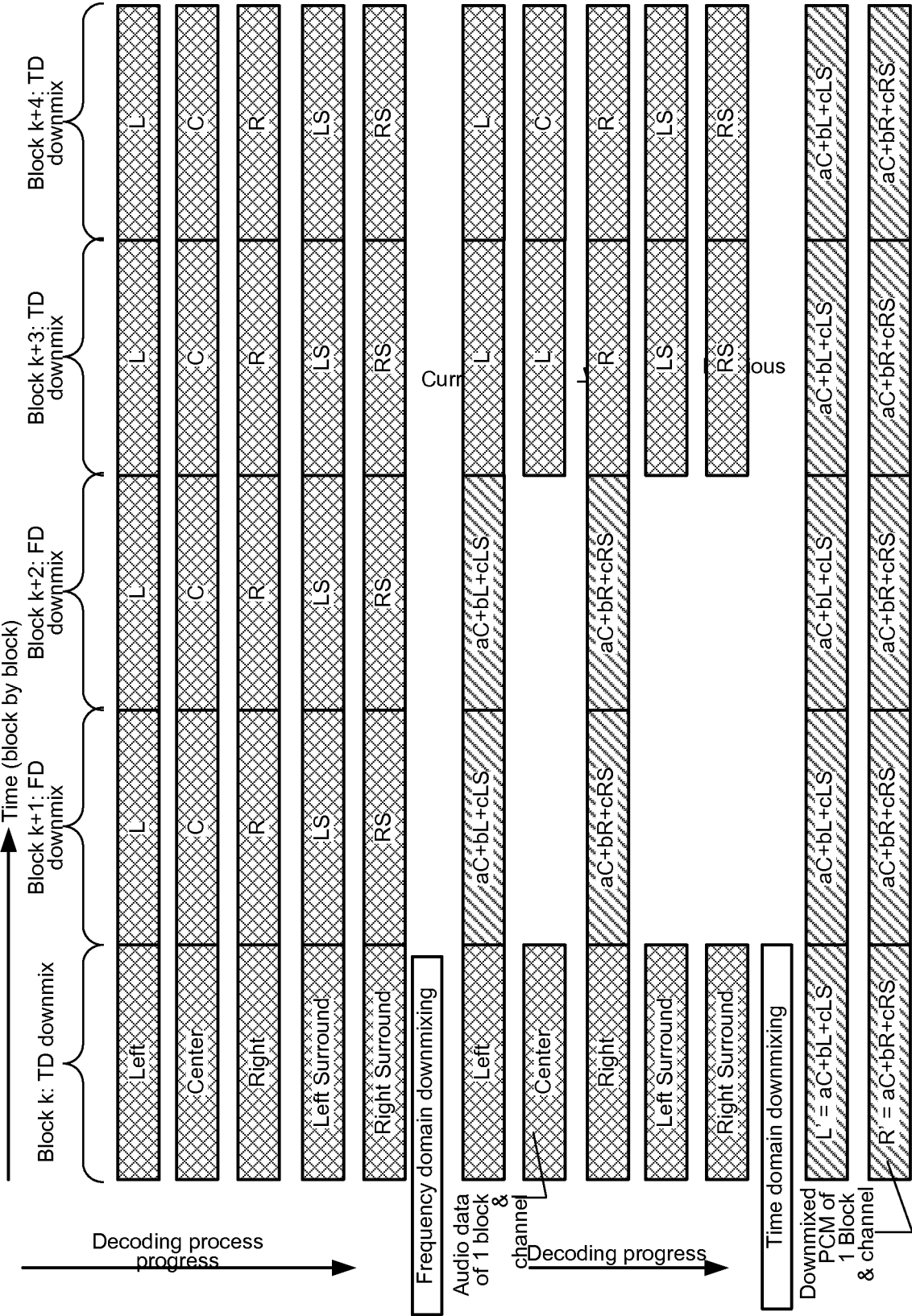


FIG. 8



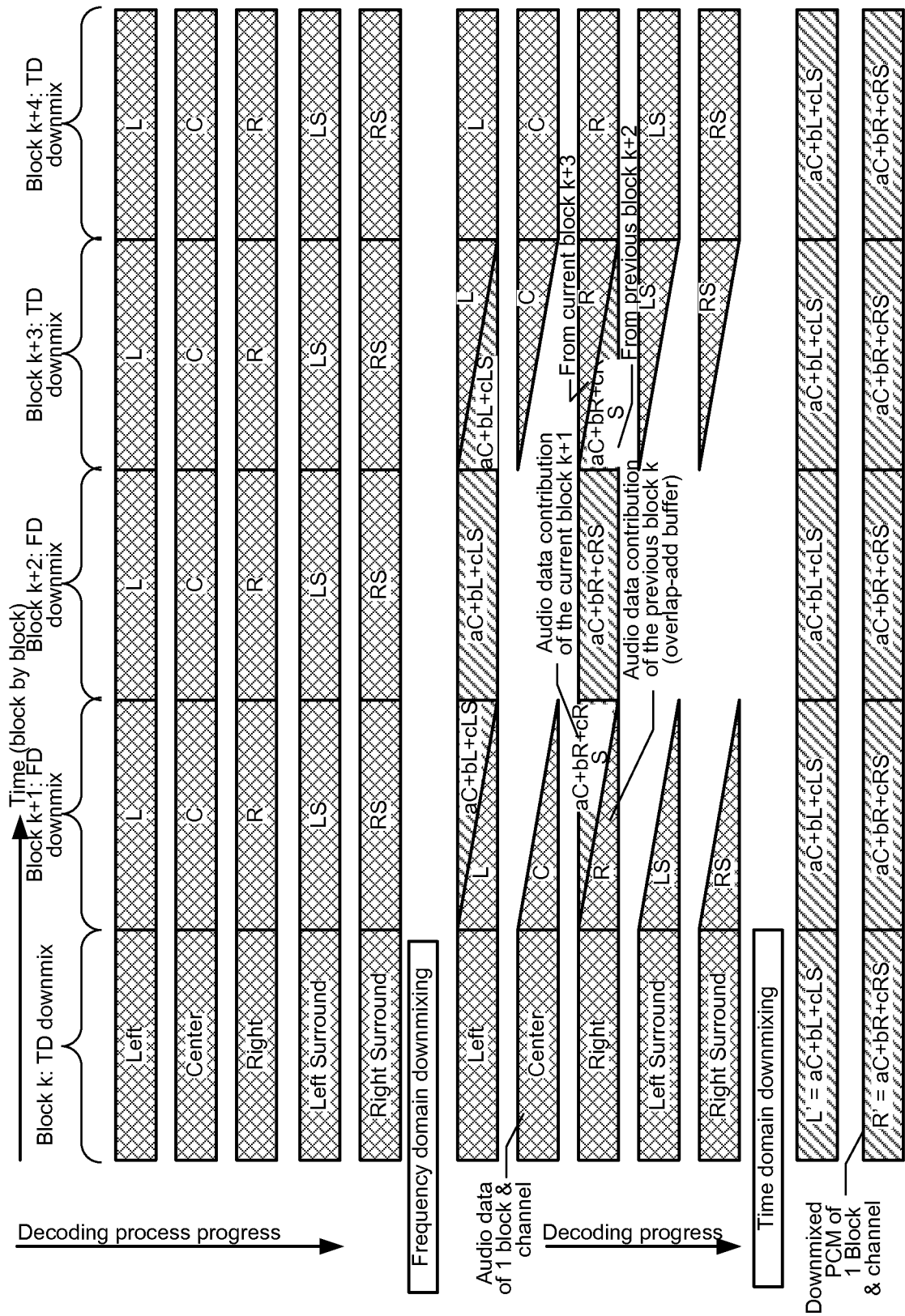


FIG. 10

12/13

```
/* TD downmix according to downmixing data*/  
if (new downmixing data = old downmixing data)  
{  
    set up for SSE instructions;  
    downmix using downmixing data;  
}  
else  
{  
    cross-fade old data to new data using a window  
    downmix using cross-faded downmixing data;  
}
```

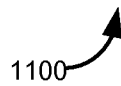
1100 

FIG. 11

13/13

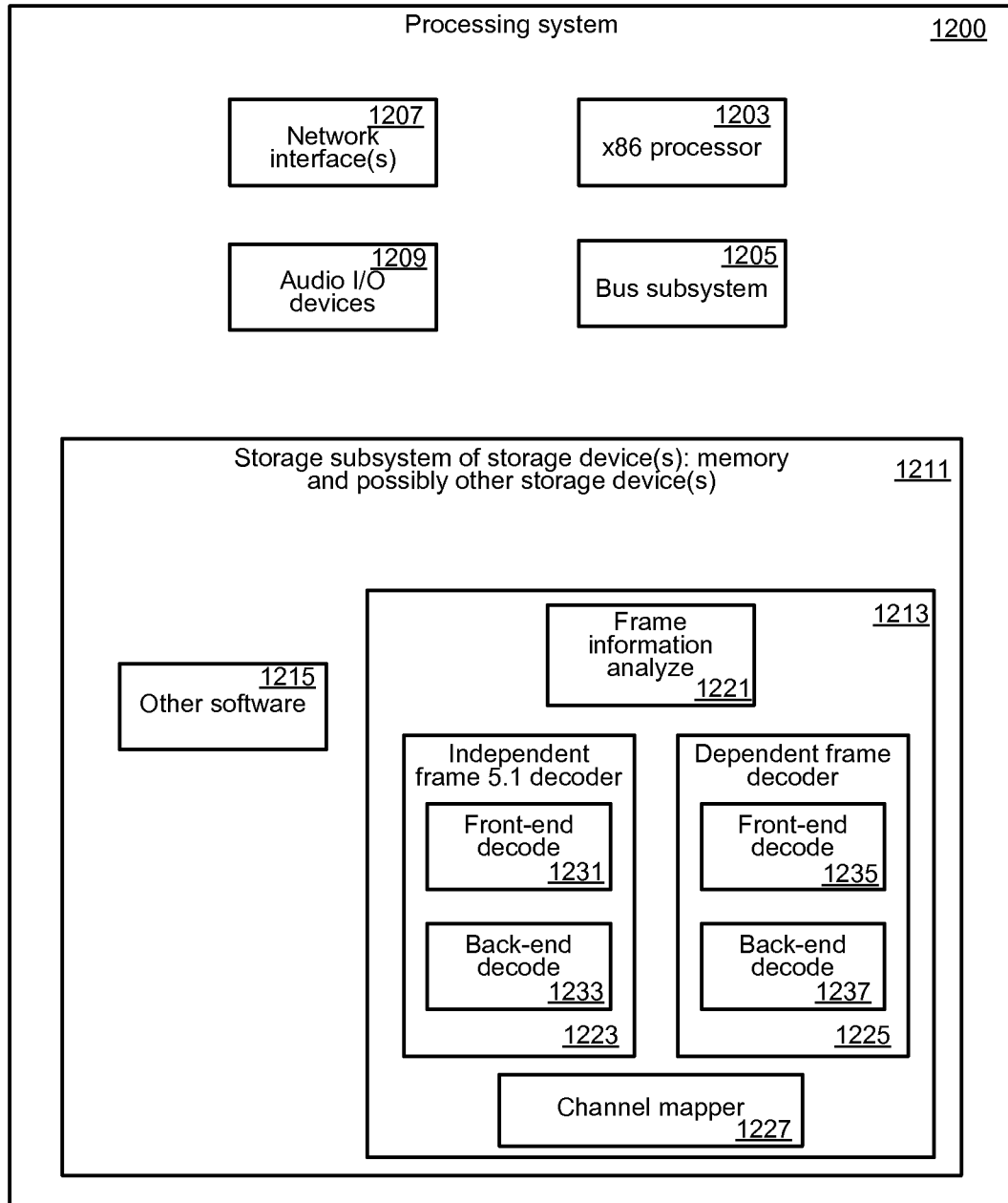


FIG. 12